UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/401,616	09/22/1999	CARL A. WALDSPURGER	9772-031-999	4899

24341 7590 09/10/2003

Pennie & Edmonds, LLP
3300 Hillview Avenue
Palo Alto, CA 94304

EXAMINER
STEELMAN, MARY J

ART UNIT	PAPER NUMBER
2122	8

DATE MAILED: 09/10/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/401,616	WALDSPURGER ET AL.	
	Examiner	Art Unit	
	Mary J. Steelman	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 July 2003* .

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-72* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-72* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *22 September 1999* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. This action is in response to Amendment A, filed 07/28/2003.

2. As per Applicant's request, the Specification has been amended.

3. As per Applicant's request, claims 1, 43, 50, 55, 66, & 69 have been amended. Claims 1-72 are pending.

Specification

4. The disclosure is objected to because of the following informalities: Page 15, line 27 recites "...the controlling tty". The meaning of 'tty' is unclear to the examiner. Tty must be defined in the Specification. In the Remarks section, page 18, of Amendment A, Applicant indicates that the common usage of the term can be confirmed via a simple search. A search done by the examiner produced four definitions. Appropriate correction is required.

5. Content of Specification: Applicant to provide Cross-References to Related Applications: See 37 CFR 1.78 and MPEP § 201.11.

Claim Objections

6. In view of the amendments to claims 50 and 55, the objections to claims 50 and 55 are hereby withdrawn.

Claim Rejections - 35 USC § 112

7. In view of the amendment to claim 43, the 35 USC 112 second paragraph rejection is hereby withdrawn.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

9. Claims 1- 12, 40, 41, 45, 54, 58 - 62, 63, 66, 69 and 70 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent 5768500 to Agrawal et al.

As to claim 1: Agrawal disclosed *"a method for value sampling...monitoring the performance of a program..."* (E.g. see col. 1, lines 55 – 58, "...method for profiling computer systems...samples system state using interrupts...") comprising the steps of:

"...executing the program on a computer system..."

"...interrupting execution...delivering a first interrupt"

"...storing at least one data value...in a ...database..."

(E.g. see Agrawal, col. 8, lines 29 – 31 where the above features are disclosed, "An interrupt handler can then record the state of the running system. If enough samples are recorded..." Examiner considers a running system to be the equivalent of executing a program and recording a sample to be equivalent to storing a value in a database.)

"...data value...associated with a particular...instruction..." (E.g. Agrawal also disclosed, col. 10, lines 26 – 34, "...system wide sampling...each time a sampling interrupt occurs, the handler appends a brief sampled state record to a kernel profiling buffer. The record typically contains information such as: (Timestamp, Current Process Id, Processor Status Word, Current Program Counter Value). Examiner considers this type of information sufficient to associate a data value with an instruction.)

"...when interrupt occurs, hwerein the particular object code instruction of the program remains unmodified..." (E.g. Agrawal, Abstract, lines 10 - 13 disclosed a method that describes "how to combine simple hardware support and sampling techniques to obtain such data without appreciably perturbing system performance." Also see Agrawal, col. 8, lines 7-14, "monitoring the system without disturbing its operation...")

As to claim 2: Agrawal also disclosed "*...intervals are random intervals.*" (E.g. see col. 16, lines 49 – 51, "It is preferred that the present invention be configured to sample at random intervals...")

As to claim 3: Agrawal also disclosed it could be possible that "*...intervals have a constant period.*" (E.g. see col. 16, lines 35 – 36, "The user programs the ranges and hardware interrupts..." Examiner considers it possible to program constant intervals, as constant intervals are merely a special case of random intervals. Also see col. 12, lines 21 – 24, "In randomized sampling...but in fixed interval sampling (*constant period*), correlations can arise producing the sort of clustering shown in Fig. 3")

As to claim 4: Agrawal also disclosed "*specifying an object code instruction*" associated with a "*data value*". (E.g. see col. 1, line 66 – col. 2, line 2, " The present invention is then able to develop a...profile that associates cache misses with specific processes...or even user defined aspects of system state." Also see col. 8, lines 43 – 45, "The initial counter value can be loaded so that an interrupt will be generated after the occurrence of a programmable number of

events..." Also see col. 9, lines 56 – 59, "...it is possible to reconfigure the monitor circuit from software, allowing us to easily reprogram the monitor (performance monitor, fig. 2, item 200) to trigger on different events or sets of events (*specify instructions*) which cause memory bottlenecks..." Also see col. 12, lines 30 – 35, "...the sampled data not only identifies which processes are missing (*data associated with instruction*)... but also which program counters are associated with these misses...Mprof records a mapping from PIDs to executable file names." Also see col. 16, lines 19 – 23, "The interrupt is then used by system software to record aspects of the program's state (*instructions and data values*)...we may customize the handler to record application specific state." The Examiner considers the above examples to relate (an association made) between a specific instruction and a data value.)

As to claim 5 and 6: Agrawal also disclosed the *"data value of interest is an operand, (or) is a result..."* (E.g. see col. 1, line 66 – col. 2, line 2, "The present invention is then able to develop a...profile that associates...user defined aspects of system state (state values)." Examiner notes that an *operand* value or a *result* value are *state* values and thus equivalent.)

As to claims 7 – 12 that relate to storing data values associated with an instruction or memory address, Agrawal also disclosed, col. 1, line 66 – col. 2, line 2, "The present invention is then able to develop a...profile that associates cache misses with specific processes, procedures, procedure call stacks or even user defined aspects of system state." Examiner considers that the user can specify any value in the system state (*instructions, kernel instructions, memory addresses, data values, program counter, current interrupt level, load and store physical*

addresses) to be associated by the monitoring process and storing process. Also see fig. 1 and col. 10, lines 9 – 12, "Upon receiving a profiling interrupt from the performance monitor of the present invention via Sbus 132 or 232, the operating system and user application cooperate with the present invention to record profile data (*store in database*)." Also see col. 10, lines 33 – 34, "A user level daemon periodically copies the profiling buffer to user space and saves its contents to a file (*store in database*)." Examiner considers this to be equivalent to saving user defined state data values (*instructions, kernel instructions, memory addresses, data values, program counter, current interrupt level, load and store physical addresses*) to a database. Also see col. 10, lines 65 – 67, "The default interface to user level sampling in Mprof requires an application to be linked with a special library (*shared library*) that replaces the default start and exit routines."

As to claims 40 and 41: Agrawal also disclosed, *"optimizing the program of...instructions based on...data value...(context information)"* (E.g. see col. 2, lines 36 – 49, "The present invention is a system and method for profiling computer system performance. The present invention may be implemented as a memory system profiler, referred to as Mprof, that samples system state *(data values, context information*) using interrupts generated by a...counter supplemented with a...register and interrupt line.... case studies are provided to illustrate how the profiles can be used to *optimize* applications..." Also see col. 8, lines 29 – 31, "An interrupt handler can then record the state of the running system (*context information*)." Examiner considers context information, as recorded in the profile, can be used to optimize the application.)

As to claim 45: Agrawal also disclosed: *"...the step of storing includes...applying a predetermined function..."* (E.g. see col. 11, lines 59 – 63, "On each profiling interrupt in system wide sampling mode, the handler logs a timestamp, the process id (PID), and the program counter (PC). Mprof uses the timestamps and PIDs to plot cache miss samples versus time (*predetermined function*) for each process.")

As to claim 54: Agrawal also disclosed: the "*step of storing includes...identifying a process identifier associated with the program and storing...(an associated) data value in memory space...* " (E.g. see col. 11, lines 59 – 63, "On each profiling interrupt in system wide sampling mode, the handler logs a timestamp, *the process id (PID) (process identifier*), and the program counter (PC*) (an associated data value*). Mprof uses the timestamps and PIDs to plot cache miss samples versus time for each process." Also see, col. 1, line 66 – col. 2, line 2, "The present invention is then able to develop a...profile that associates cache misses with specific processes (*process identifier*), procedures, procedure call stacks or even user defined aspects of system state." Also see fig. 1 and col. 10, lines 9 – 12, "Upon receiving a profiling interrupt from the performance monitor of the present invention via Sbus 132 or 232, the operating system and user application cooperate with the present invention to record profile data (*data value of interest associated with process identifier*)." Also see col. 10, lines 33 – 34, "A user level daemon periodically copies the profiling buffer to user space and saves its contents to a file (*store in a memory space*)." Examiner considers the contents saved to a file to be accessible to the program.)

As to claim 58: Agrawal also disclosed, "*storing...most frequently occurring data values...*" (E.g. see col. 12, lines 11 – 13; "...over time (*frequency*), sampling (*for data values*) correctly attributes misses to the processes that are incurring them..." Also col. 12, lines 30 – 32, "...the sampled data not only identifies which processes (*data values*)...but also which program counters (*data values*) are associated..."

"*predetermined number of data values...*" See col. 9, lines 56 – 59, "...it is possible to reconfigure the monitor circuit from software, allowing us to easily reprogram the monitor to trigger on different events (*predetermine number of data values*) or sets of events which cause memory bottlenecks..."

"*...and a count associated with each value...*" See col. 11, lines 59 – 65, "On each profiling interrupt in system wide sampling mode, the handler logs a timestamp, the process id (PID), and the program counter (PC). Mprof uses the timestamps and PIDs to plot cache miss samples versus time for each process. (count vs. time) Fig. 3 displays a plot for our test workload...line 302 corresponds to the misses generated by the single invocation..."

Also see col. 14, lines 49 – 52; "By also recording the virtual address of the cache miss that triggers the sampling interrupt, one could get a crude profile of hot spots (*hotlist of most frequently occurring data values*) in the data.")

As to claims 59 – 61: Agrawal also disclosed, "*encoding the hotlist (frequent values)*"; "*sort...data values...*"; "*reorder...data values...such that...values...are stored in contiguous memory locations...*"(E.g. see col. 7. lines 8 – 10, "The program counter counts are then post-

processed by prof to produce a table of per procedure execution times." Also see col. 10, lines 9 – 12, "Upon receiving a profiling interrupt from the performance monitor of the present invention...the operating system and user application cooperate with the present invention to record profile data." Also see col. 10, lines 33 – 42, "A user-level daemon periodically copies the profiling buffer to user space and saves its contents to a file. Post mortem tools can then synthesize cache miss profiles...reporting user and system mode cache misses and plotting cache miss samples versus time. In addition...symbol tables may be used to synthesize...profiles that give ...counts..." Also see col. 12, lines 10 – 11, "Mprof does...help us to visualize and quantify the effect." Also see col. 12, lines 35 – 41, "Mprof records...During the post mortem data viewing phase, when the user clicks on one of the lines of the summary graph...Mprof exploits this record to map...into an executable file name, extracts the symbol table from the file and then synthesizes a ...profile from its sampled program counter information." Also see col. 12, lines 44 – 48, " ...Mprof's ability to relate data stall cycles to call paths...compares several approaches to sorting...records..." Also see col. 14, lines 50 – 52; "...recording the virtual address of the cache miss that triggers the sampling interrupt, one could get a crude profile of hot spots (*encoding a hotlist*) in the data." Examiner considers that functionally, Mprof, as disclosed by Agrawal, has the ability to count, sort, reorder, the data encoded after profiling, including 'hot' data values, when in the post mortem processing stage.)

As to claims 62 and 63: Agrawal also disclosed, *"randomized techniques"*. (E.g. see col. 16, lines 47 – 51, "The present invention simply interrupts the program every time N units of resource are accumulated (by a counter) and records the program state. It is preferred that the

present invention be configured to sample at random intervals with mean N units to avoid correlation problems." Also see col. 2, lines 39 – 41, "...Mprof, that samples system state using interrupts generated by a loadable decrementer...counter supplemented with a compare register and interrupt line..." Also see col. 8, lines 16 – 19, "...multiple counters *(first randomized technique and a second randomized technique*) accumulate occurrences of waits in specified time ranges and interrupt the processor when one of the counts reaches a specified value.")

As to claim 66: This is a system version of the claimed limitations disclosed in claim 1 above. Additionally Agrawal discloses a *system* and a *processor*. (E.g. see col. 1, lines 55 – 65, "The present invention is a *system*...for profiling computer systems...samples system state using *interrupts* generated...One advantage of the present invention is that it has the ability to interrupt the *processor*...This improvement...enables the present invention to sample the state of the *processor*..."

As to claim 69: This is a computer program product version of the claimed limitations disclosed in claim 1 above, where all claim limitations have been addressed.

As to claim 70: This is a computer program product version as addressed in claim 69. Furthermore the "*random interval*" claim limitation has been addressed in claim 2 above.

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) do not apply to the examination of this application as the application being examined was not (1) filed on or after November 29, 2000, or (2) voluntarily published under 35 U.S.C. 122(b). Therefore, this application is examined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

11. Claims 1-10, 12-14, 22-41, 45-47, 51-54, 66, 68, 69 and 72 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent 6195748, to Chrysos et al.

As to claim 1: Chrysos disclosed "*a method for...sampling...monitoring the performance...comprising*": (E.g. see Abstract, line 1-2, "sampling instruction in a processor pipeline..." also see col. 10, lines 7, 8, "Profiling hardware...gathers detailed state information." And col. 8, line 31, "It is desired to collect detailed performance data..." Also see col. 6, lines 49 – 53, "Specifically, provided is an apparatus and method for periodically, and randomly selecting one or more instructions processed by a pipeline of a processor, and to collect profile information while the instruction progresses...")

"*executing...instructions...*" (E.g. see col. 9, lines 4 – 5, "...instructions are fetched during a single processor cycle." Examiner considers this to be equivalent to executing instructions. Also see col. 17, lines 29 – 38, "...profiling can include the following steps...an

instruction is fetched...the instruction is selected...As the selected instruction progresses through the execution pipeline, profile information is collected...")

"...at intervals interrupting execution..." (E.g. See col. 13, lines 42, 43, "The profile register file can be read when the fields have been updated and the interrupt signal is generated." Also see col. 14, lines 54-55, "The size of the interval determines the average frequency of sampling.")

"...storing...data value...in a ...database..." (E.g. see col. 6, lines 51 – 56, "...collect profile information while the instruction progresses through stages of an execution pipeline. Higher level software can then post process this information in a variety of ways, such as by aggregating information from multiple executions of the same instruction.") *"data value...associated with a particular...instruction..."* (E.g. see col. 8, lines 32, 33, "...collect detailed performance data while the instructions execute. Performance data can be related to memory operations and execution flows." Also see, col. 10, lines 11,12, "The state information can be directly attributed to specific events.") *"...when the interrupt occurs, wherein particular object code instruction of the program remains unmodified..."* (E.g. see col. 1, line 66 – col. 2, line 4, "Hardware implemented event sampling can also be used to provide profile information of processors. Hardware sampling has a number of advantages...: it does not require modifying software programs to measure their performance. Sampling works on complete systems...")

As to claims 2 and 3: Chrysos disclosed a privileged profiling software component (PSW) that could be used to initialize a fetch counter, to set intervals to "*random intervals*" or "*constant periods*". (E.g. see col. 14, lines 46 – 57, "As shown in Fig. 5 for a fetch stage

implementation...a fetch counter 510 is initialized by...privileged profiling software (PSW) 520...The PSW 520 can initialize the counter 510 with a value randomly selected from an interval of values having a predetermined size...The size of the interval determines the average frequency of sampling. Other randomizing techniques to initialize the value of counter 510, including hardware, may also be used." Examiner considers the PSW could be programmed for any time interval, and a "*constant period*" is merely a variation of a "*random interval*", thus equivalent.)

As to claim 4: Chrysos disclosed the step of "*specifying an...instruction...wherein the data value...is associated with the...instruction...*" (E.g. see col. 11, lines 48 – 54, "...performance information related to the selected instruction can be captured...the registers collect performance information *(data values of interest)* that is attributable to specific known instructions and events.")

As to claims 5 and 6: Chrysos disclosed the "*data value of interest is an operand*" or the "*result of the execution of the instruction.*" (E.g. See Abstract, line 6, "...state information of the system is sampled while any of the selected instructions are in any stage of the pipeline..." Operands and results are state information. Additionally see col. 29, lines 14 – 18, "...state information includes operand values of the selected instructions...and intermediate values (*results*) computed by the selected instructions.")

As to claims 7 - 9: Chrysos disclosed the *"associated memory address and...data value"*, an *"associated data value and instruction derived from a shared library"*, and an *"associated kernel instruction..."* are stored in a database. (E.g. see col. 8, lines 22 – 27, "During operation of the system (*kernel instructions*), instructions and data of software programs (*shared libraries*) are stored in the memories. The instructions and data are generated...using compiler, linker and loader techniques. The instructions and data are transferred to the execution pipeline of one of the processors." Also see col. 8, lines 31 – 33; "It is desired to collect detailed performance data (*data values*) while the instructions execute. Performance data can be related to memory operations and execution flows." Also see col. 10, lines 7 – 12, "Profiling hardware...gathers detailed state information. The state information can come from any of the pipeline stages, or elsewhere in the system...other sub systems. The state information can be directly attributed to specific events." Also see fig. 3 and col. 11, lines 39 – 41, "...there is provided a memory 300 for storing profile information for each instruction being sampled..." Also see col. 11, lines 52 – 54, "...the registers collect performance information that is attributable to specific known instructions and events." Also see, col. 12, lines 12 – 17, "A profile effective address register is loaded with an address associated with the selected instruction. If the instruction is a memory access instruction, such as a load or store, then the ...memory address is captured. If the instruction is a jump or branch, then the target PC is recorded. (*memory address stored in a program counter*)" Also see col. 16, lines 23 – 25, "The sample information can be stored in a memory buffer (*database*) for later processing by profiling tools to produce detailed instruction level information." Also see col. 17, lines 40 – 41,

"Sampled information can be buffered for subsequent processing." Examiner considers this to be equivalent to storing data values in a database.)

As to claim 10: Chrysos also disclosed *"specifying a set of ...instructions...wherein storing is performed...when the program is interrupted at a point associated with...instructions..."* (E.g. see figs. 8A and 9, and col. 19, lines 52 – 60, "...a reasonable size of the window W, 820, around instruction I, 810, should include about a hundred instructions...Given a window of size W...unbiased sampling can be performed by randomizing the fetch distance between the selected instructions." Examiner interprets this to mean that a set of instructions is selected and interrupts capture sampling data on the set of instructions.)

As to claim 12: Chrysos also disclosed, *"storing...physical addresses for load and store instructions..."* (E.g. see col. 10, lines 7 – 8, "Profiling hardware dynamically gathers detailed state information. Also see col. 29, lines 16 – 26, "...state information includes effective addresses...effective addresses are physical addresses of instructions. (*load and store instructions*).")

As to claims 13 and 14: Chrysos also disclosed, *"...storing...data value that identifies the destination of...instruction that transfers control"* and specifically the *"instruction that transfers control flow...*(is a) *(conditional / unconditional branch instruction, jump instruction, or call / return instruction)"* (E.g. see col. 10, lines 7 – 8, "Profiling hardware dynamically gathers detailed state information." Also see col. 29, lines 16 – 29, "...state information includes

effective addresses and intermediate values computed...effective addresses are virtual memory addresses of data...of instructions...are physical addresses of data...of instructions...state information includes history information about taken / not-taken status of recently executed branch instructions..." Examiner considers the branch instructions and their associated addresses to be equivalent to *identification of destination of instruction that transfers control.*)

As to claims 22 and 24: Chrysos also disclosed, *"...storing the memory address and the interrupted instruction"* (E.g. see col. 10, lines 7 – 8, "Profiling hardware dynamically gathers detailed state information." Examiner considers the memory address and the instruction to be part of state information.)

"configuring a second interrupt...delivered after a predetermined number of instructions are executed..." (E.g. see col. 18, lines 10 – 15, "The profiling technique described herein can also be used to perform N-wise sampling. Here, the dynamic state of interactions between multiple concurrently executing instructions can be captured. Instead of profiling a single in-flight instruction, two or more separate instructions are concurrently profiled. (*requiring a second interrupt*) Also see col. 19, lines 8 – 33, "...the privileged profiling software (PSW) can dynamically vary the size of the interval from which the initial values of the two fetch counters are randomly selected (program the PSW to sample after a predetermined number of instruction are executed). This allows the inter-sample fetch distance for the pair of instruction to be specified at the same time...Because N-wise sampling involved two levels of sampling...")

"...deactivating the second interrupt..." (E.g. see fig. 5, the PSW is programmed to control the interrupts.)

"...storing the memory address and...data value..." "for the associated instruction" and "*storing the interrupted instruction*" Storing the memory address, data value and associated instruction, and interrupted instruction are equivalent to storing state information and have been discussed above in claim 7.

As to claims 23 and 27: Chrysos also disclosed that a "*predetermined number of instructions*" could be "*equal to a number of instructions of an issue block.*" (E.g. see col. 9, lines 4 – 7, "Preferably, a set of 'instructions' are fetched during a single processor cycle. The set can include, for example, four instructions. In other words, the pipeline is four slots wide." In reference to claim 27, a "*set of...instructions*" fetched for a processor cycle is equivalent to an "*issue block.*")

As to claim 25: Chrysos also disclosed, *"analyzing the interrupted instruction to determine the...data value of interest to store..."* (E.g. see col. 10, lines 7 – 12, "Profiling hardware dynamically gathers detailed state information. The state information can come from any of the pipeline stages, or elsewhere in the system...The state information can be directly attributed to specific events." Also see fig. 7A and col. 17, lines 53 – 67, "...function 755 takes as input state information 756...of the selected instructions (*analyzing*). Step 760 produces a subset of samples based on the function 755. In step 780, statistics 790 are determined. These statistics can include averages...of the properties of the sampled instructions. Also see col. 17, lines 38 – 41, "...profile information is collected in step 740. Upon completion...the collected

information is sampled in step 750. Sampled information can be buffered for subsequent processing. (*to store*)")

As to claim 26: Chrysos also disclosed, an *"instruction...associated with a set of...instructions...including an interrupted instruction...associated with a memory address..."* See col. 6, lines 35 – 41, "...implement a nested form of sampling...a window of instructions (*set of instructions*) that may execute concurrently with a first profiled instruction is dynamically defined. A second instruction is randomly selected for profiling from the window of instructions. The profiled *(interrupted instruction*) and second instruction form a sample pair for which profile information (from a second interrupt) can be collected."

Additionally, the following limitations have been addressed in claims 22 and 24 above: *"storing the memory address and the set of ...instructions...", "configuring a second interrupt...", "deactivating the second interrupt...", "storing the memory address and...data value..."* Examiner considers memory addresses, data values and instructions to be included in state information that is profiled and stored.

As to claims 28 and 29: Chrysos also disclosed, *"the predetermined number of events is a predetermined number of instruction executions."* (E.g. see col. 10, lines 11 and 12, "The state information can be directly attributed to specific events. Also see fig. 2B and col. 10, lines 31 and 32, The selection is controlled by a value of a counter, 510. The value of the counter can be initialized by line (init) 511." Also see col. 13, lines 38 – 40, "...each instruction fetched can be consecutively numbered with an 'inum' value. In this case, instructions with specific inum

values can be selected (predetermine number of instruction executions)." Also see fig. 5 and col. 15, line 43, "What is selected can be filtered by a filter 505."

Additionally, in reference to claim 29, where the "*predetermined number of events is a predetermined number of clock cycles*", see col. 15, lines 11 – 14, "the counter 510 is incremented every cycle, instead of for each instruction fetched...")

As to claims 30 and 31: Chrysos also disclosed, "*storing...data value...of the first database in a second database*" and "*generating...(a) value profile for ...data value in second database.*" (E.g. see fig. 7B where items 770, 780, and 790 could represent a first and second form of database and furthermore a histogram or property statistic could represent a *value profile* for a *data value.* Also see col. 17, line 63 – col. 18, line 5, "Step 760 produces a subset of samples based on the function 755. In step 780, statistics 790 are determined. These statistics can include averages...of the sampled instructions...The histograms can show the distribution of instruction execution...")

As to claims 32 – 35: Chrysos also disclosed, the "*...data value...has a plurality of data values of interest, and...stores a tuple of the...data values and a memory address...*" (E.g. see fig. 4 and col. 13, lines 49 – 58, "...the augmented instruction can include the following additional fields, up to three instruction operands..., the program counter, the operator code (*a plurality of data values of interest)*...Fields 440 and 450 can be reserved for indicating instruction related I-cache and TLB misses...Note because a single instruction can include multiple operands..." Also see col.16, lines 62 – 67, "Filtering of the profiling information can

also be done by hardware means...only sample...other...combinations of opcodes, operands, addresses, events and latencies." Examiner considers grouping of multiple data values of interest collected from profiling, are associated into a tuple, a set of elements, for storing.)

As to claim 36: Chrysos also disclosed an "*issue block of instructions...*" "*storing...data values...including a value of a register, a return address register, and a value stored in a memory location in a...stack frame.*" (E.g. see col. 9, lines 4 – 6, "a set of 'instructions' (*issue block*) are fetched during a single processor cycle. The set can include,...four instructions." Also see col. 24, lines 55 – 62, "The classes of instruction can be identified as...conditional branch, call, return, (*profile value in a return address register*) access...The class can be selected by a selection mechanism...The class can...be identified by the stage of the pipeline...The class ID may be controlled by software." Examiner considers that profiling can collect information on any state information and functionally a register holds data similar to the way a stack frame holds data.)

As to claim 37: Chrysos also disclosed, a "*data value...and the tuple*" whereby the tuple has "*a count...associated with...data values*" and "*storing...includes...identifying the...tuple...based on the ...data value...*" and "*incrementing the count associated with the...tuple when the...data value...is the same...*" (E.g. see col. 12, lines 8 – 12, "...the profile information can be directly attributed to a specific instruction. In addition, the sampled information can be filtered according to range of addresses, opcode, execution threads, address spaces, and the like." Also see col. 16, lines 23 – 30, "The sample information can be stored in a

memory buffer for later processing by profiling tools to produce detailed instruction-level information. The information can be used to develop...histograms...perhaps even moderately comprehensive analysis of the pipeline state for each instruction." Examiner considers that profile information could be stored in tuple form as discussed in claim 34, above, profiling tools could be used to produce aggregated results of stored data, and specifically a histogram would require a count feature.)

As to claims 38 and 39: Chrysos also disclosed multiple state values that could be profiled and stored. The "*program counter*", a related "*invoked function containing the profiled instruction*", a "*destination register value*", and a "*return address which invoked a function*" are examples of state information that could be extracted by profiling. Values can be stored as "correlated". (E.g. see fig. 7B and col. 17, lines 53 – 64; "...function 755 takes as input state information 756 such as addresses, process identifiers, address space numbers...of the selected instructions. Function 755 may also use state information such as path-identifying information, opcodes, operands...experienced by the selected instructions...Step 760 produces a subset of samples (*correlated)* based on the function 755. In step 780, statistics 790 are determined." Another example of an invoked function can be found at col. 21, lines 41 – 49, "A subtractor decrements the number of instruction retired...from the register...the count stored in the head entry of the FIFO queue. The output of the register is divided by the number (P) of the cycles tracked to yield the dynamic or instantaneous average number...The instantaneous average may be captured in the profile registers, or stored in a processor register, or memory location readable by software.")

As to claims 40 and 41: Chrysos also disclosed "*optimizing...based on...data value*" and the data value could be "*context information.*" (E.g. see col. 10, lines 13 – 19; "...the design strategy is to collect information (data values) that is difficult to determine statically in a profile record. This makes the profile record useful for performance tools, profile-directed optimization, or for making resource allocation policy decisions in operating systems and application level software, including dynamic adjustments directly in response to the sampling and analysis. Also see col. 28, lines 24 – 25, "...the state information (data value) includes a hardware context identifier.")

As to claims 45, 46, and 47: Chrysos also disclosed, *"applying a predetermined function...before data value is stored..."* (E.g. see fig. 7B and col. 17, lines 53 – 67 as discussed above in claims 38 and 39.)

"identifying an instruction...(and) operand...and...storing..." (E.g. see discussion on specifying and instruction and operand in claims 5 and 6 above.)

"*generating...projected value...by applying a function...and storing...*" (E.g. see fig. 7B and col. 17, lines 53 – 67 as discussed above in claims 38 and 39.)

"...data value...and...tuple..." "...tuple stores a functional value..." "...database stores a set of tuples..." "updating the functional value in the...tuple..." (E.g. in reference to state information stored in tuple form, see discussion for claims 32 – 35. As to updating the tuple, see 7B and col. 17, lines 63 – 65, "Step 760 produces a subset of samples based on the function

755." Examiner considers any state value that is processed by a statistical function to produce a subset to be equivalent.)

As to claims 51 - 54: Chrysos also disclosed, "*storing includes...delivering...data value...to interrupted program.*" (E.g. a data value can be any state value, including a result needed to continue execution of the program.)

"*...user-mode interrupt to deliver...data value...*" (E.g. see col. 6, line 38, "a first profiled instruction is dynamically defined." Examiner considers defining an instruction to be an example of a user mode interrupt that will store a data value sample.)

"*...instructions are executed in a kernel...upcall from the kernel to a user-mode handler...*" (E.g. see col. 21, lines 47 – 49, "The instantaneous average may be captured in the profile registers (*executed in a kernel*), or stored in a processor register, or memory location readable by software (*to a user-mode handler*)." Also see col. 10, lines 31 – 32, ""The selection is controlled (*executed in a kernel*) by a value of a counter.")

"*...identifying a process identifier...*" (E.g. see col. 12, lines 1 – 2, "Other fields of the register can store the *process identifier*...")

"*...storing...data value...in...memory...associated with the process identifier...*" (E.g. see col. 12, lines 8 – 12, "By storing this information, the profile information can be directly attributed to a specific instruction...the sampled information can be filtered according to...addresses, opcode, ...threads...and the like." Examiner considers this equivalent to storing a selection of state values, including the process identifier, for further processing.)

As to claim 66: This is a system version of the claimed limitations discussed above in claim 1. Additional claim limitations, "*a processor for executing...a memory for storing instructions...*" are also disclosed by Chrysos. (E.g. see fig. 1 and col. 8, lines 4 – 7, "...computer system which can use the sampling method and apparatus as described herein. The system includes one or more processors 110, off-chip memories 120, ...")

As to claim 69: This is a computer program product version of the claimed limitations discussed above in claim 1. Also see col. 30, lines 20 – 23, "...a computer readable medium (*computer program product*) embodying program code configured to cause a computer to process the sampled information associated with the selected instructions including aggregating..."

As to claims 68 and 72: Chrysos also disclosed, a system and computer program product including "...*a second interrupt*..." which is delivered after a predetermined number of instructions. The second interrupt is deactivated and data values of interest are stored. See col. 6, lines 34 – 42, "...in order to measure useful concurrency, a technique called 'pair-wise sampling" is provided. The basic idea is to implement a nested form of sampling...a first profiled instruction is dynamically defined. A second instruction is randomly selected for profiling from the window of instructions. The profiled and second instruction form a sample pair for which profile information can be collected." Also see col. 20, lines 29 – 30, " The profile data recorded (when *interrupt activated and deactivated*) for each pair-wise sample...includes values stored (*store data values*)..." Examiner considers data values to include

all state variables, including addresses and association with instructions. Activation and deactivation of interrupts is displayed in fig. 2 B, note fetch, 210, map, 220, issue, 230, execute, 240, retire, 250, then interrupt, 541.

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claim 15 is rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6195748 to Chrysos et al., in view of U.S. Patent No. 6237073 to Dean et al.

Chrysos disclosed interrupt hardware sampling. Chrysos did not disclose a data value identifying the destination of a conditional branch instruction, where a bit identifies the destination. Dean disclosed random sampling of state information with branch history bits are specified as one of the data values sampled. See fig. 2B, item 282 and col. 10, lines 40 – 41, "State information, such as instruction addresses...branch history bits (HIST) 282...can be sampled on lines 288."

It would have been obvious, to one having ordinary skill in the art at the time the invention was made to modify the hardware interrupt sampling, as taught by Chrysos, by specifying destination of a conditional branch with identification by a bit, and post processing

sampled data values, as taught by Dean, because the destination could be a data value of interest (a state value) in performance monitoring and specifying a (branch destination) taken/not taken bit would help classify the data for post mortem statistical processing.

14. Claims 16 – 21, 67 and 71 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5768500, to Agrawal et al., in view of U. S. Patent No. 6192516, to Griesemer. Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose, "*interpreting.*" However, Griesemer disclosed interpreter generation and implementation utilizing interpreter states and register caching. Griesemer disclosed the following:

"identifying and interpreting the instructions of the ...issue block of instructions...and storing ...data value...associated with...instruction..." See fig. 11 and col. 14, lines 53 - 55, "Once the selected virtual machine instruction has been executed (*interpreting the instructions*), the system fetches (*identifying and retrieving an issue block of instructions*) the next virtual machine instruction at step 1105. Also see fig. 8

"...interpreting emulates a machine language instruction set ..." See col. 1, lines 33 – 38, "The Java virtual machine is a software emulator of a "generic" computer...The Java virtual machine is commonly implemented as a software interpreter." Examiner considers this to be equivalent to an interpreter emulating machine language.

"...interpreter updates a state of the interrupted program..." See col. 2, lines 36 – 40; "If the selected state differs from the state of the interpreter that is expected...computer code for the interpreter is generated to put the interpreter in the expected state."

"...interpreter interprets...kernel and user code..." See col. 1, lines 51 – 53, "...interpreter must be executing in order to decode and execute an interpreted program (*user code*), the software interpreter consumes resources (e.g., memory)...(*kernel code*)"

"...interpreter does not execute particular instructions." See col. 11, lines 66 – col. 12, line 2, "...if it is determined that the selected virtual machine instruction and interpreter state are illegal, the system may generate computer code to handle the error..." Examiner considers this to mean that certain illegal code is not executed.

It would have been obvious, to one having ordinary skill in the art, at the time the invention was made, to modify performance monitoring, as taught by Agrawal, by monitoring with an interpreter utilizing states and register caching, as taught by Griesemer, because many programs currently are in a platform neutral code form which allows for execution on any computer that is able to run a virtual machine interpreter. Thus, using an interpreter to process instructions allows a larger number of programs to be sampled.

15. Claims 42 – 44 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5768500, to Agrawal et al., in view of U. S. Patent No. 6237073 to Dean et al.

Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose "*storing comprising the steps of generating another program*"... "*when...data values...(exceed) a predetermined / sufficient amount of invariance*", *including references to memory addresses, prefetching, and expected vs. actual values*. Dean did disclose a data sample file that generates another program. State values are sampled (*data values of interest*) and include a wide range of values. See fig. 7B and col. 18, lines 11 – 40, "As shown in FIG. 7b, a

process 799 estimates statistics of properties of instructions processed by the pipeline 200. The process 700 can include the following steps. Step 751 reads the profile record...The record is read when the selected instruction completes. In step 760, the sample is selected or discarded depending on a function 755 (*generate another program)*, which takes into consideration state information (*includes memory addresses*) of the system.

For example, function 755 takes as input state information 756 such as addresses, process identifiers, address space numbers, hardware context identifiers, or thread identifiers of the selected instructions. Function 755 may also use state information, such as ...event experienced by the selected instructions.

Step 760 produces a subset of samples (comparison of actual vs. expected) based on the function 755. In step 789, statistics 790 are determined. These statistics can include averages, standard deviations, histograms...The bound on the errors can be approximated..."

Examiner considers that function 755 can address levels of invariance, prefetching instructions and actual vs. expected values, as these are considered state values. Function 755 can produce a wide variety of statistical analysis (compare actual vs. expected). Thus, Examiner considers this to be equivalent to the claim limitations.

It would have been obvious, to one having ordinary skill in the art, at the time the invention was made, to modify the performance monitoring method, as taught by Agrawal, by generating another program to process data values in various statistical procedures, as taught by Dean, because post sampling processing for statistical purposes would enable the performance monitoring results to be summarized and quantified and thus be more useful for determining

"how new computer hardware operates with existing operating systems and application programs" (Applicant's Background of the Invention, lines 8 and 9).

16. Claims 48 – 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5768500, to Agrawal et al., in view of U.S. Patent No. 6192516 to Griesemer, and further in view of U. S. Patent No. 6233600, to Salas et al. Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose interpreting instructions of an issue block, and storing an associated data value. However, Griesemer disclosed interpreter generation and implementation utilizing interpreter states and register caching. Neither Agrawal nor Griesemer disclosed receiving a downloaded script, executing with a virtual machine, or user-mode traps. However, Salas did disclose these features in his method and system for providing a networked collaborative work environment.

Salas disclosed a method to (col. 2, lines 8 - 22) "download files *(downloaded script)*...allow a set of geographically...separate users to participate...receiving data associated with a project (*receiving an interpretable program*)...*storing* the received project data..." Also see col. 1, lines 31 – 33, "The browser executing on the client workstation interprets (*interpreting...the instructions*) the HTML file that it has received and displays (*executing ...instructions with a virtual machine*) the Web page to the user..." Also see col. 5, line 31 – 35, "The embedded discussion 420 and the contribution facility 422 may be implemented as...a JAVA applet (*bytecode instructions*)..." Also see col. 5, lines 55 – 57, "The graphical element 406 may be static...or it may be a dynamic identifier such as a *JAVA script* (*bytecode*

instructions) ..." Also see col. 12, lines 15 – 17, "File download and subsequent upload, if necessary, is managed by a background daemon *(user-mode trap)*."

It would have been obvious, to one having ordinary skill in the art, at the time the invention was made, to modify the performance monitoring method, as taught by Agrawal, with interpreter generation and implementation utilizing interpreter states and register caching, as taught by Griesemer, and further modify the invention with a script download, interpreted with a virtual machine, as disclosed by Salas, because downloading interpretable script would allow performance monitoring programs to be run on a wide variety of computing platforms using a byte code instruction set.

17. Claims 55 - 57 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5768500, to Agrawal et al., in view of U. S. Patent No. 6233600, to Salas et al. Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose access control identifiers, or identifiers for a process, user, or group. However, Salas did disclose these features in his method and system for providing a networked collaborative work environment.

See col. 14, lines 38 – 54, "...access of users to each item can be controlled by entries in the database schema...a list (*access control identifiers*) of the members that are entitled to enter...users may be divided into...groups (*user identifier / group identifier*)...User access (*particular user*) may be checked by running the database query...and only allow a user to access (*only a user associated with the group identifier can access*)...when that user's name or authentications context appear as an entry in the table..." Also see fig. 7 where access is

controlled at item 706. Also see col. 13, lines 31 – 55, "Access rights may be checked...fields which identify users (*access control identifiers*) ...Alternatively, an object may assign a pre-defined value to a field which controls access to the object...File metadata (*process identifier*) may include: ...access information..."

It would have been obvious, to one having ordinary skill in the art, at the time the invention was made, to modify the performance monitoring method, as taught by Agrawal, with access controls, as disclosed by Salas, because access controls would provide security in collaborative work environments.

18. Claims 64 and 65 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5768500, to Agrawal et al., in view of Gibbons, P. B., et al., "New Sampling-Based Summary Statistics for Improving Approximate Query Answers," Proc. ACM SIGMOD, 1998.

Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose "*concise samples technique*" nor "*counting samples technique*". However, Gibbons did disclose (page 332, left column, section 1.1) "...two new sampling-based summary statistics, concise samples and counting samples..."

It would have been obvious, to one having ordinary skill in the art at the time the invention was made to modify the performance monitoring method, as taught by Agrawal, with the "*concise samples technique*" and "*counting samples technique*" when (Agrawal, col. 14, lines 33) "synthesizing different types of profile...and relating program data..." because concise samples and counting samples, both sampling-based summary statistics, (Gibbons, page 332,

right col.) "provide more sample points for the same footprint, they provide more accurate estimations."

Response to Arguments

19. Applicant's arguments filed in Amendment A, 05/28/2003 have been fully considered but they are not persuasive. Applicant argues, in substance:

(A) Since data is sampled when the interrupt occurs, and since execution of the interrupted instruction has not yet been completed, the data is not associated with the particular object code instruction of the program being executed by the computer when the interrupt occurs. Agrawal's system associates the sampled data with the instruction that initiates the interrupt as opposed to the instruction being executed when interrupt actually occurs.

Examiner responds: Agrawal, col. 10, lines 22-32, "...performance monitor...supports ...data collection...each time a sampling interrupt occurs, the handler appends a...record to a...profiling buffer. The record typically contains information such as: (Timestamp, Current Process ID, Processor Status Word, Current Program Counter Value). Applicant, for an example, explains that a program counter would not accurately identify the instruction executed and thus associated with the sampled data. However, Examiner believes that using a timestamp to record which instruction executed would correctly identify the instruction executed and associate it with the sampled data. Note Agrawal's preferred approach, col. 10, lines 45-64, "...application opens a special sampling device to indicate to the interrupt handler...that it wishes to receive memory system sampling signals. When a sampling interrupt occurs, the interrupt handler checks whether the currently executing process has opened the sampling device...posts a signal to the process...kernel...restarts the process in the signal handler that the

user process has registered for the sampling signal (in effect, similar to Applicant's executing the instruction via interpreter or via a second interrupt). This handler records relevant aspects of the process state and then control returns to the process' previously executing thread of execution."

(B) Agrawal fails to teach, "...wherein the particular object code instruction of the program remains unmodified."

Examiner responds: Agrawal does not indicate that the instruction is modified, only that profiling is done through interrupts, and data is recorded. Agrawal, col. 8, line 9, "...monitoring the system without disturbing its operation..."

(C) Chrysos presents a system having additional hardware apparatus, whereas Applicant's invention does not require augmenting system hardware.

Examiner responds: This is not a claim limitation.

Conclusion

20. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the vent a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

21. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (703) 305-4564. The examiner can normally be reached Monday through Thursday and alternate Fridays, from 7:15 A.M. to 4:45 P.M. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on (703) 305-4522.

20. The fax phone number is (703) 872-9306. Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Mary Steelman

09/03/2003

TUAN DAM
SUPERVISORY PATENT EXAMINER